UML-Based Approach to Developing Verified Embedded Software

Sara Van Langenhove

Prof. Dr. A. Hoogewijs

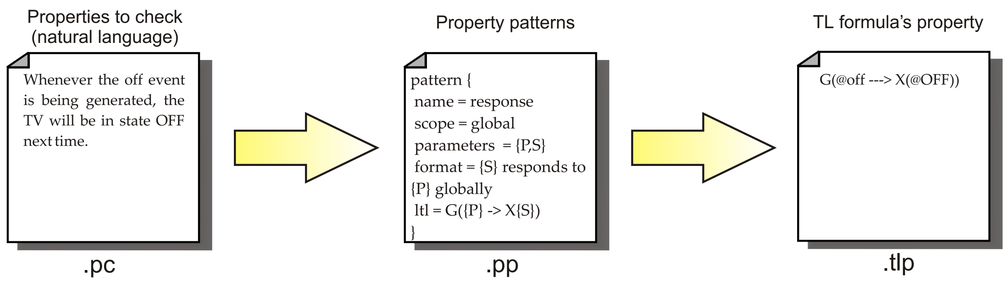
Abstract

In today's world, embedded systems are everywhere: from alarm systems to x-ray machines and everywhere in between. If you start to look around you, you are bound to realize that you are surrounded by devices with computing power inside. Embedded systems have become a vital part of a growing range of automotive, aerospace, biomedical and military systems. Concurrently, embedded systems have become much more complex, in part because more powerful processors are now inexpensive enough to be included in cars, cameras, kid toys and other products. These embedded devices, with more processing power and memory, typically perform a complex set of functions, or even several functions. As many of these applications are potentially life-threatening, the need for an appropriate design approach has never been more compelling.

The central aim of embedded systems design is to develop concepts, methods and tools that can master their growing complexity and at the same time reduce development costs and time to market. In fact, it is becoming a trend to use the Unified Modeling Language (UML) to design embedded software.

We are convinced that software errors are likely to be introduced in the design phase and that these errors have a lasting impact on the reliability, cost and safety of a system. We propose a UML-based verification method to identify and remove errors, misconceptions, etc. during the design phase of embedded software development.

Requirements

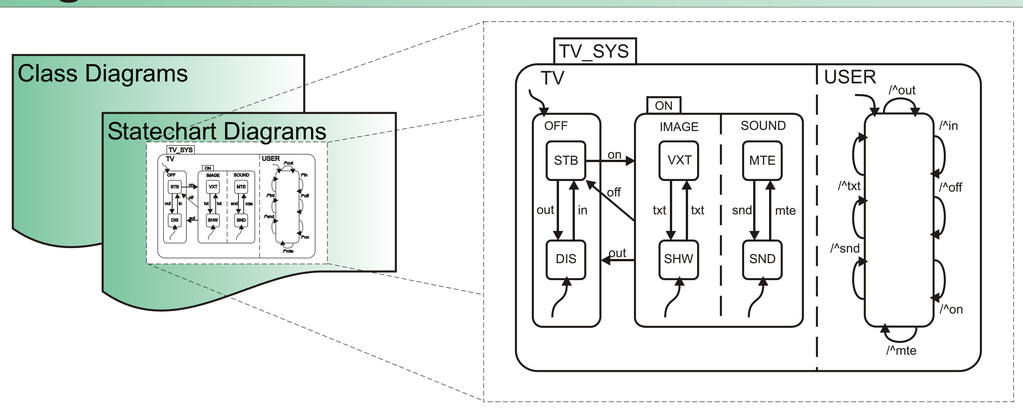


Requirements Analysis involves describing important properties, in English, about (embedded) software systems with the intention of specifying what the eventual system will be expected to provide. Since English is often imprecise and ambiguous, temporal logic has been proposed, as an alternative.

Quite often the requirements of a system follow simple patterns. A property specification pattern describes the essential structure (= scope) of some aspect of a system's behavior. E.g. the scope "global" means that the requirement should always hold.

Using such property patterns, expectation phrases are transformed into temporal logic formulae, through an assistant that guides the user in writing the properties.

Design

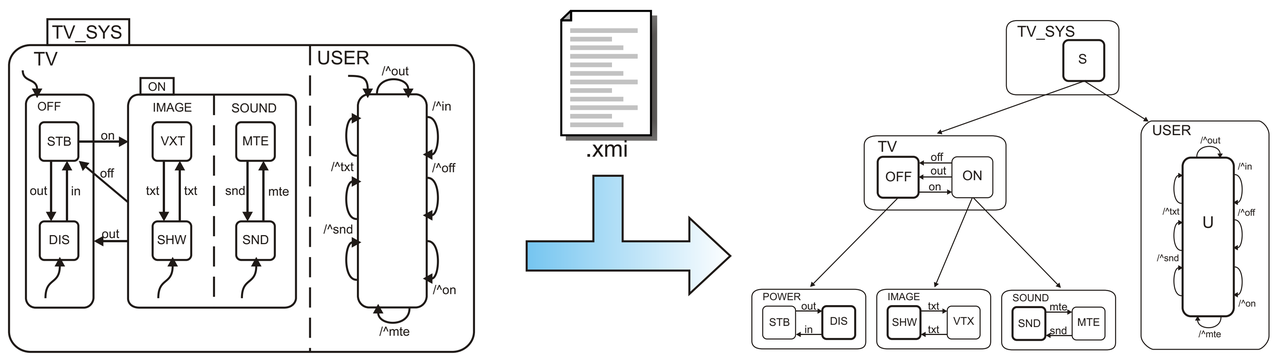


A design is a preliminary sketch. It is an outline of the main features of the system's components or how the system is supposed to execute. Several diagrams are used for this activity.

Like any other embedded system, a TV is event-driven, meaning that it continuously wait for the occurrence of some external or internal event (power on or power off). After recognizing the event, such systems react by performing the appropriate computation such as changing to the appropriate state e.g. switch to teletext. Once the event handling is complete, the software goes back to a dormant state in anticipation of the next event. This reactive system behavior is described by graphical statechart diagrams.

Verification

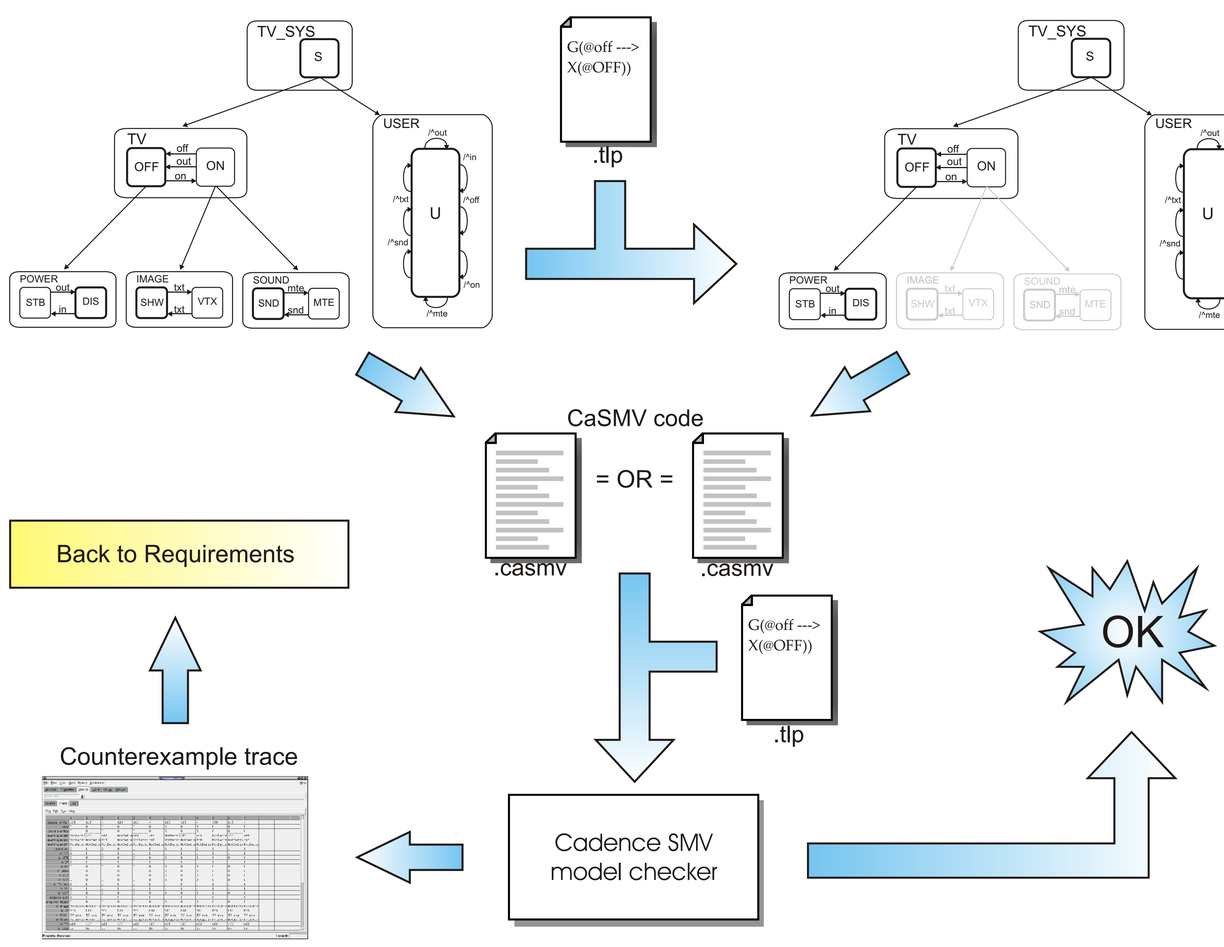
a) Transformation from State Chart to Extended Hierarchical Automaton (EHA)



The input is a UML statechart specification which has been formatted using the XMI (XML Metadata Interchange) exchange syntax. With no intervention of the user, the system's behavior is automatically transformed, through XMI, into an Extended Hierarchical Automaton, which is an alternate equivalent representation of the system's behavior. An EHA is built as parallel and/or hierarchical composition of sequential automata whose states themselves can be other automata.

E.g. state Off is refined into a single sequential automaton, whereas state On is composed of two concurrent sequential automata.

b) Verifying embedded behavior through model checking and with(out) slicing



Symbolic model checking is a powerful formal verification technique for reactive systems, but sometimes slicing can be necessary for improving the time and space efficiency of symbolic model checking for systems specified as state charts. Using a property, slicing removes those parts of the EHA irrelevant for verifying the property.

E.g. to prove the given property, it is not necessary to also consider the automata Image/Sound during verification.

From EHAs, a formal representation in CaSMV is automatically obtained.

CaSMV (Cadence Symbolic Model Verifier) uses symbolic model checking for the verification. This means that the test is automatic, always obtains an answer and more importantly, should the property not be satisfied, it generates a means of identifying the originating error (counterexample).

This research is part of a large cooperation between theoretical computer scientists and engineers. It is funded by "Geconcentreerde Onderzoeksacties" (GOA/BOF B/03667/01 IV1) and the Prof. Wuytack Fund of the UGent. This research focuses mainly on interdisciplinary cooperation using the right communication tools, in this case UML.